BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHQTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(12) UK Patent Application (19) GB (11) 2 324 939 (13) A

(43) Date of A Publication 04.11.1998

(21) Application No 9806180.7

(22) Date of Filing 23.03.1998

(30) Priority Data
(31) 08846357 (32) 30.04.1997 (33) US

(71) Applicant(s)
Hewlett-Packard Company
(Incorporated in USA - Delaware)
3000 Hanover Street, Palo Alto,
California 94303-0890, United States of America

(72) Inventor(s)
Daniel E Yee
Robert W Cherry
Byron A Alcorn

(74) Agent and/or Address for Service
Carpmaels & Ransford
43 Bloomsbury Square, LONDON, WC1A 2RA,
United Kingdom

(51) INT CL[6]
H04N 13/00

(52) UK CL (Edition P)
H4T TBAS T131
H4F FDD FD12X FD15 FD30K

(56) Documents Cited
GB 2312122 A GB 2308284 A

(58) Field of Search
UK CL (Edition P) H4F FDD, H4T TBAS
INT CL[6] G06T 13/00 17/00, H04N 13/00
Online database: WPI

(54) Abstract Title
Stereoscopic image signal production using single image buffer

(57) System for generating left and right video channels to drive a stereoscopic display device 44 using only one frame buffer memory 16 of a computer graphics pipeline, comprises window circuitry 22 for defining first and second windows 52, 54 within the frame buffer 16. The first and second windows 52, 54 store pixel data corresponding to left and right views, respectively, of a stereoscopic image. A stereo output system is operable to present pixel data read from the first window 52 within the frame buffer memory 16 to the pixel data input 36 of a first video encoder system 32, and to present pixel data read from the second window 54 within the frame buffer 16 memory to the pixel data input 38 of a second video encoder system 34. Video format signals that are output by the first and second video encoder systems 32, 34 comprise the left and right video channels. Position and size information are stored within a memory device 22. As pixel data is read serially out of the frame buffer memory 16, a count is maintained to keep track of the coordinates of the pixels. The coordinates for a given pixel are compared with the stored position and size information. The pixel is output to left or right video encoders 32, 34 if it is determined that the pixel corresponds to the first or second window 52, 54, respectively.

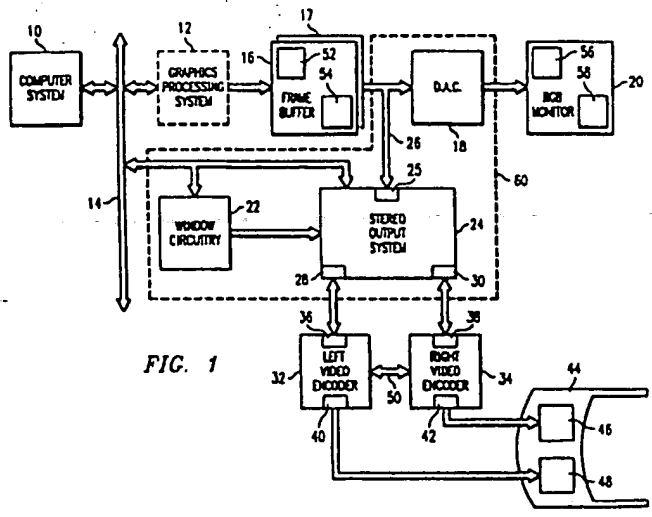


FIG. 1

GB 2 324 939 A

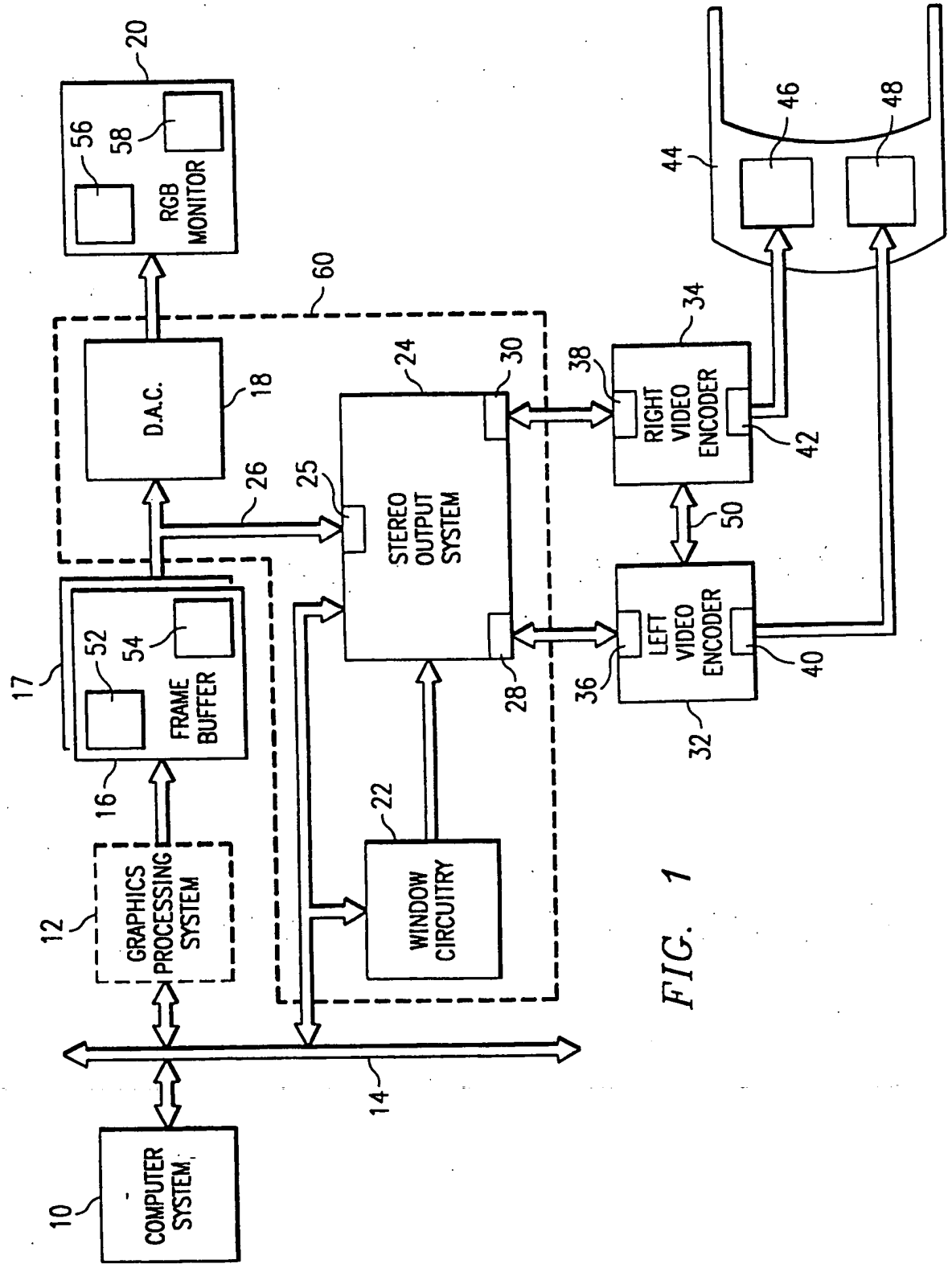

COMPUTER SYSTEM
10
GRAPHICS PROCESSING SYSTEM
12
14
FRAME BUFFER
16
17
52
54
WINDOW CIRCUITRY
22
STEREO OUTPUT SYSTEM
24
25
26
28
30
D.A.C.
18
RGB MONITOR
20
56
58
60
RIGHT VIDEO ENCODER
34
36
38
LEFT VIDEO ENCODER
32
40
42
50
44
46
48

FIG. 1

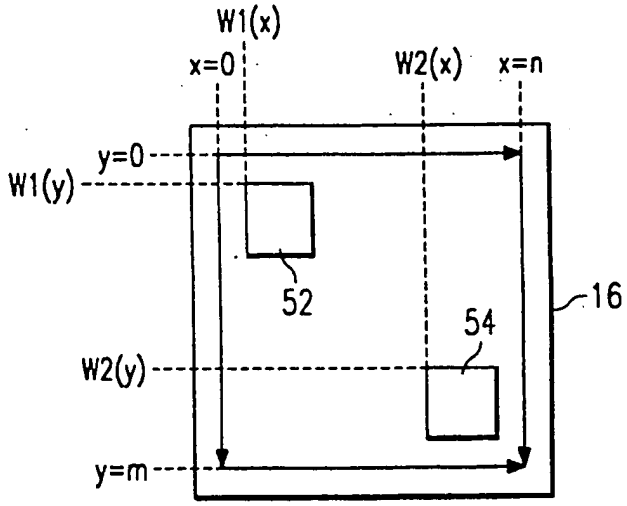

W1(x)
x=0
W2(x)
x=n
y=0
W1(y)
52
54
16
W2(y)
y=m

FIG. 2

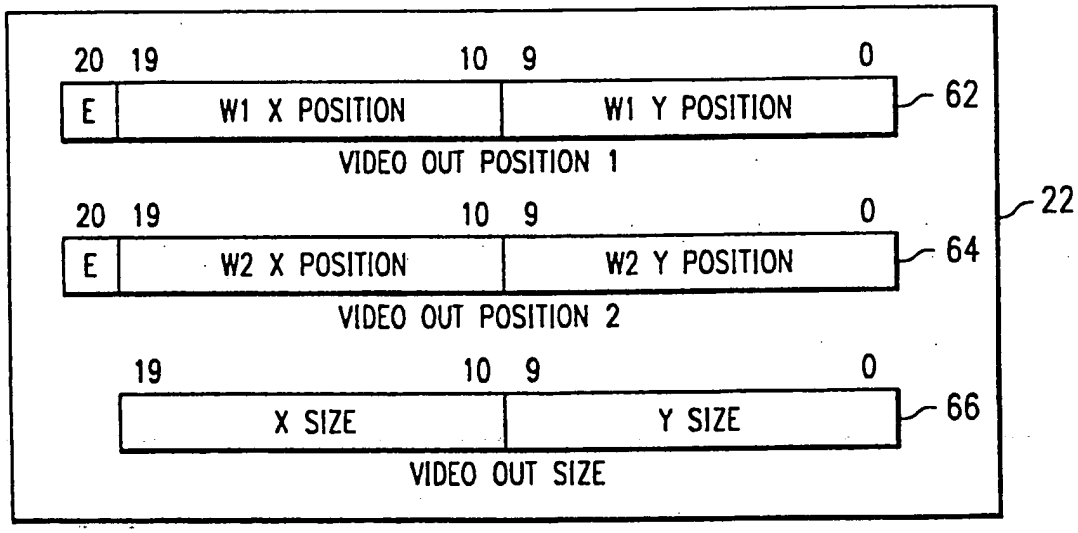

20 19
10 9
0
E
W1 X POSITION
W1 Y POSITION
62
VIDEO OUT POSITION 1
20 19
10 9
0
E
W2 X POSITION
W2 Y POSITION
64
VIDEO OUT POSITION 2
22
19
10 9
0
X SIZE
Y SIZE
66
VIDEO OUT SIZE

FIG. 3

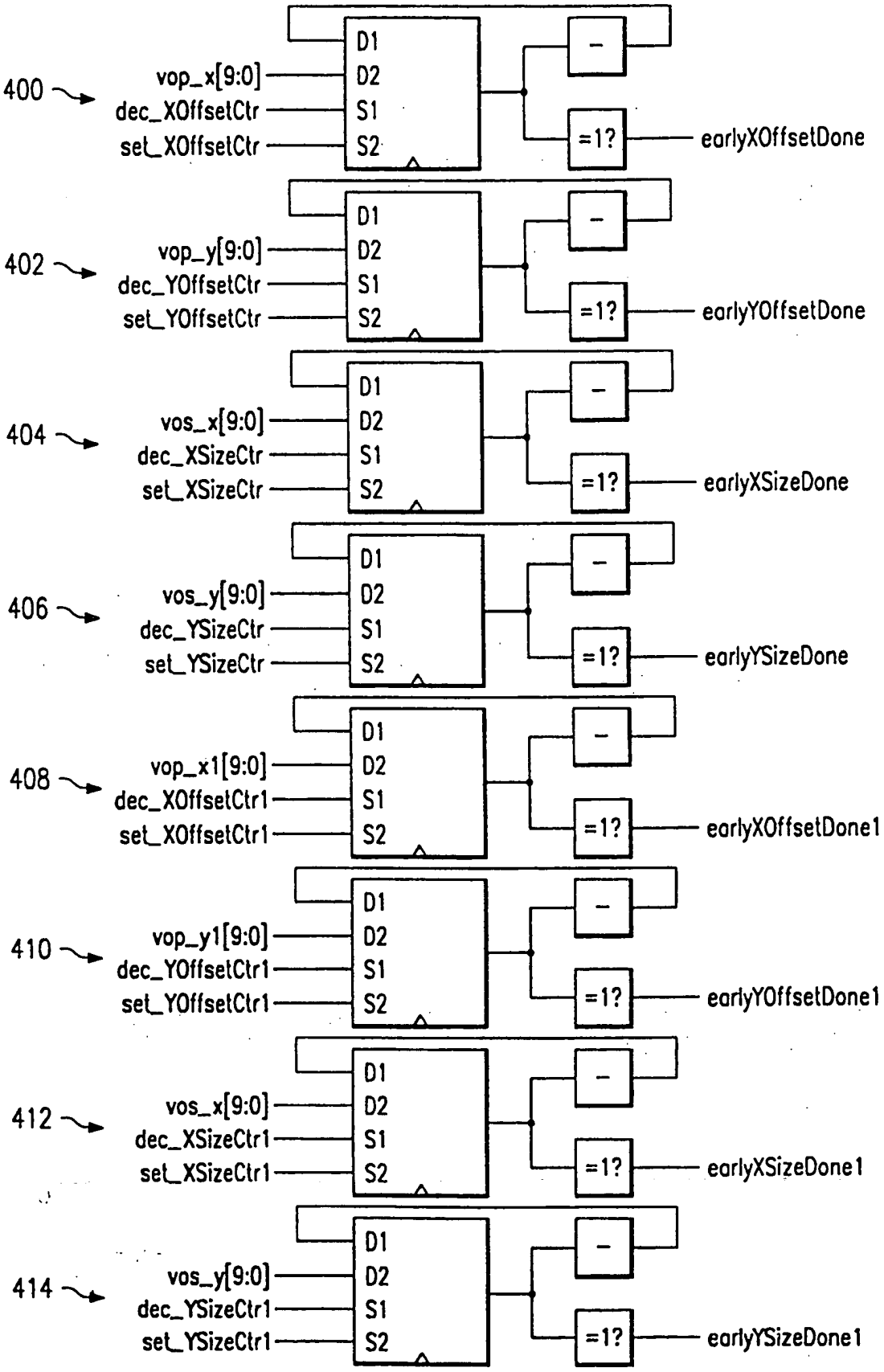

400
vop_x[9:0]
dec_XOffsetCtr
set_XOffsetCtr
D1
D2
S1
S2
=1?
earlyXOffsetDone
402
vop_y[9:0]
dec_YOffsetCtr
set_YOffsetCtr
earlyYOffsetDone
404
vos_x[9:0]
dec_XSizeCtr
set_XSizeCtr
earlyXSizeDone
406
vos_y[9:0]
dec_YSizeCtr
set_YSizeCtr
earlyYSizeDone
408
vop_x1[9:0]
dec_XOffsetCtr1
set_XOffsetCtr1
earlyXOffsetDone1
410
vop_y1[9:0]
dec_YOffsetCtr1
set_YOffsetCtr1
earlyYOffsetDone1
412
vos_x[9:0]
dec_XSizeCtr1
set_XSizeCtr1
earlyXSizeDone1
414
vos_y[9:0]
dec_YSizeCtr1
set_YSizeCtr1
earlyYSizeDone1

FIG. 4

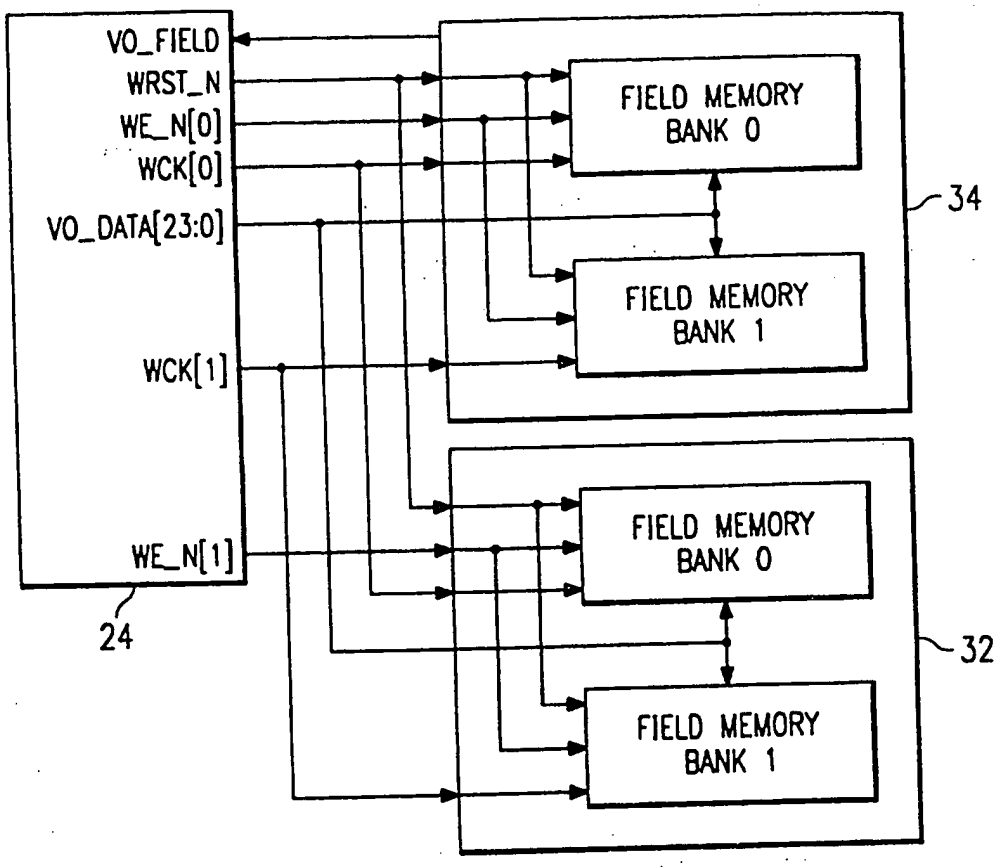


FIG. 5

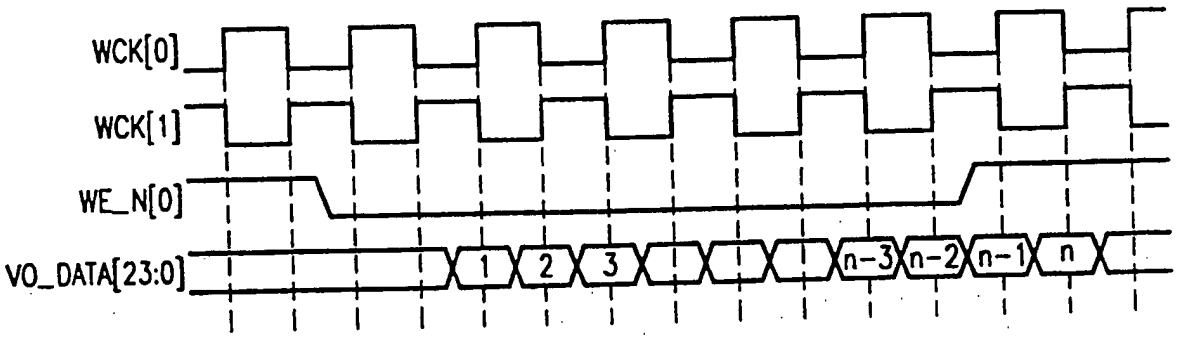


FIG. 6

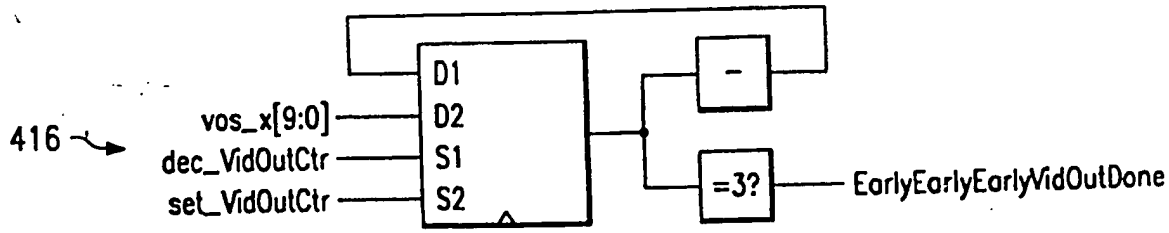
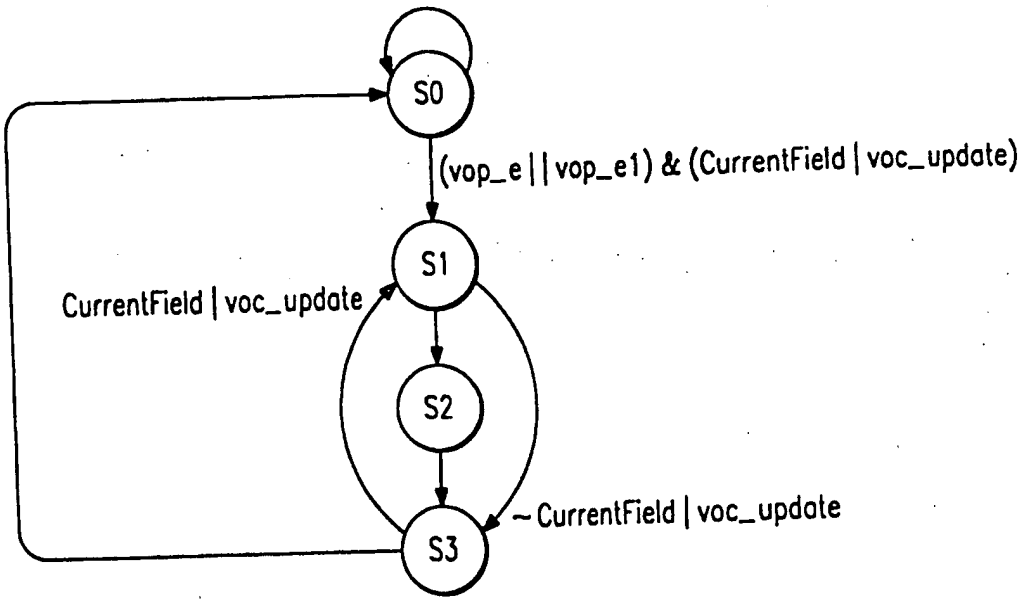


FIG. 7

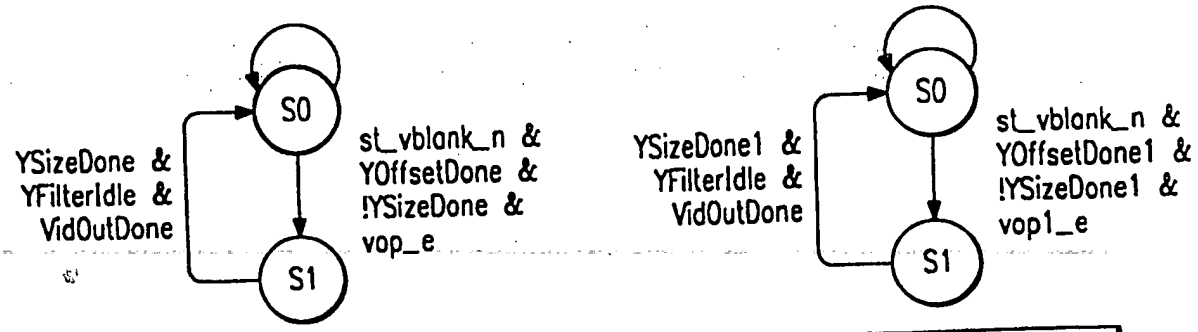


STATE	OutputField	OutputEnable	st_wrst_n
S0	0	0	1
S1	0	1	0
S2	0	0	1
S3	1	1	1

NOTE: st_wrst_n IS SET TO 1 ON POSEDGE OF hblank_n

S0 – WAIT FOR DISPLAY OF ODD VIDEO FIELD
S1 – OUTPUT EVEN FIELD
S2 – WAIT FOR DISPLAY OF EVEN VIDEO FIELD
S3 – OUPUT ODD FIELD

FIG. 8

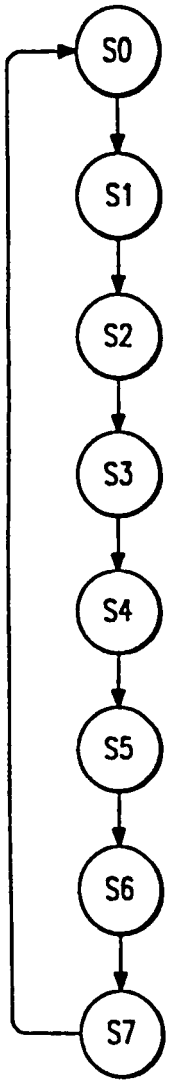


STATE	WindowState0
S0	1
S1	0

S0 – IDLE
S1 – ENABLE VIDEO OUPUT

STATE	WindowState1
S0	1
S1	0

FIG. 10



STATE	pre_st_wck[1:0]	WriteEnable_n	RequestVideoPixel	sampled_WriteEnable_n	crc_enable
S0	10	NOTE 1	0	NOTE 1	0
S1	01	NOTE 1	!WriteEnable_n	NOTE 1	0
S2	10	NOTE 2	0	WriteEnable_n	0
S3	01	NOTE 1	0	NOTE 1	NOTE 4
S4	10	NOTE 1	0	NOTE 1	0
S5	01	NOTE 1	!sample_WriteEnable_n	NOTE 1	0
S6	10	NOTE 3	0	NOTE 1	0
S7	01	NOTE 1	0	NOTE 1	NOTE 4

NOTE 1: HOLD LAST VALUE

NOTE 2: IF (EarlyEarlyVidOutDone)
WriteEnable_n=1
ELSE IF (VideoOutReady && !VidOutDone)
WriteEnable_n=0

NOTE 3: IF (EarlyEarlyVidOutDone)
WriteEnable_n=1

NOTE 4: IF (!sampled_WriteEnable_n && EnableCRC)
crc_enable = 1;

FIG. 9

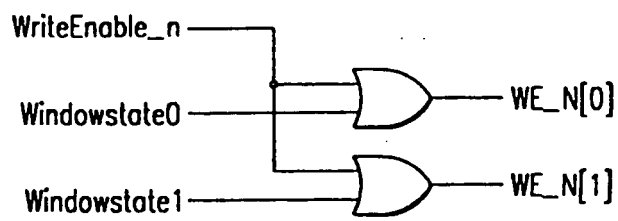


FIG. 11

System and Method for Generating Stereoscopic Display Signals from a Single Computer Graphics Pipeline

Field of the Invention

This invention relates generally to computer display systems, and more particularly to computer display systems that present images stereoscopically.

Background

Stereoscopic display methods enable the observer to perceive depth in an image. This result is accomplished by presenting two different views of the same scene independently, one to each of the observer's eyes. If the viewpoints of the two images are separated in space by a distance similar to the distance between human eyes, then the effect for the observer is a very convincing visual experience of three dimensions.

Various kinds of stereoscopic display methods have been employed to produce this effect. One common method uses a single display with special eyewear that changes the displayed image differently for each of the observer's eyes. Examples of such eyewear include glasses whose lenses comprise different color filters, different polarization filters, or shutters that alternately occlude the views of the observer's left and right eyes in synchronization with a changing displayed image. See, for example, U.S. patent no. 5,572,250, issued to Lipton et al., titled "Universal Electronic Stereoscopic Display." When used in low-end systems, such glasses sometimes introduce unwanted visual artifacts such as flickering or blurring into the observer's perception of the image.

Another known stereoscopic display method uses separate displays for each of the observer's eyes. The latter method inherently eliminates much of the flicker and blur associated with the special eyewear required in single-display systems. On the other hand, dual-display systems can be prohibitively expensive because of the duplication of hardware required to generate separate but simultaneous left and right images.

By way of example, U.S. patent no. 5,488,952, issued to Schoolman, titled "Stereoscopically Display Three Dimensional Ultrasound Imaging" discloses a system having

not only two displays but also two display drivers, two graphics controllers, two imaging computers and two image memories. Similarly, U.S. patent no. 5,493,595, also issued to Schoolman, discloses a system having two displays and two display processors. In both of these systems, polygon data is generated independently for the left and right images. Then, duplicate left and right hardware pipelines are used separately and simultaneously to rasterize and display the left and right polygon data. Thus, these designs require dual frame buffers and a commensurate added expense.

By way of further example, U.S. patent no. 5,523,886, issued to Johnson-Williams et al., titled "Stereoscopic/Monoscopic Video Display System" discloses a system having two separately-addressable frame buffers and two video-format displays. One of the two frame buffers contains rasterized pixel data corresponding to the left image. The other frame buffer contains rasterized pixel data corresponding to the right image. A single video encoder is used to convert the left and right pixel data to video format signals in time-division-multiplexed fashion. The two display devices are alternately enabled and disabled in synchronization with the alternating left and right signals output by the video encoder. Such a system not only adds the expense of dual frame buffers, but also potentially adds flicker to the image and causes fatigue to the viewer by alternating the duty cycle of the left and right displays. Johnson-Williams alternatively suggests, at col. 3, lines 24-27, that a single frame buffer may be used with the just-described system "if the video signal source can change or exchange the data in the single [frame buffer] fast enough to provide a left view or a right view when needed by [the video encoder]." Thus, in the single frame buffer embodiment taught by Johnson-Williams, left and right views are alternated in the frame buffer. Such a solution would reduce the potential refresh of the left and right displays by 50%, once again potentially causing flicker and viewer fatigue.

It is believed that another example of a solution in which separate frame buffers are used to store left and right images can be found in two products sold by Silicon Graphics, Inc., under the trademarks ONYX and CRIMSON. Once again, such solutions are expensive to the extent that they require duplicate frame buffer hardware for the left and right channels.

It is an object of the present invention to provide a system and method for generating left and right stereoscopic images from a computer graphics system that utilizes only a single frame buffer and a single conventional graphics pipeline.

It is a further object of the present invention to allow simultaneous and continuous viewing by both eyes without alternating the presentation of the left and right views or reducing the normal refresh rate of the display devices presenting the views.

Summary of the Invention

In one embodiment, the invention includes a system for generating left and right video channels to drive a stereoscopic display device using only one frame buffer memory of a computer graphics pipeline. Window circuitry is provided for defining first and second windows within the frame buffer memory. The first and second windows store pixel data corresponding to left and right views, respectively, of a stereoscopic image. First and second video encoder systems are also provided. Each video encoder system is operable to output a video format signal responsive to a pixel data input. A stereo output system is coupled to the frame buffer memory, the window circuitry and the first and second video encoder systems. The stereo output system is operable to present pixel data read from the first window within the frame buffer memory to the pixel data input of the first video encoder system, and to present pixel data read from the second window within the frame buffer memory to the pixel data input of the second video encoder system. In this manner, the video format signals that are output by the first and second video encoder systems represent the left and right video channels.

In another embodiment, the invention includes a method for generating left and right video channels to drive a stereoscopic display device using only one frame buffer memory of a computer graphics pipeline. First and second windows are allocated within the frame buffer memory, and pixel data corresponding to a left and a right image are stored within the first and second windows, respectively. Also, position and size information are stored within a memory device. The position and size information correspond to the position and size of the first and

second allocated windows. As pixel data is read serially out of the frame buffer memory, an X count and a Y count is maintained so as to keep track of the coordinates of the pixels as they are read out of the frame buffer memory. The X count and the Y count for a given pixel are compared with the stored position and size information. Based on the outcome of this comparison, a determination is made as to whether the given pixel corresponds to the first window, the second window, or neither of the windows. In further embodiments, if it is determined that the given pixel corresponds to the first window, then pixel data corresponding to the given pixel is output to a first video encoder. Otherwise, if it is determined that the given pixel corresponds to the second window, then pixel data corresponding to the given pixel is output to a second video encoder. If it is determined that the given pixel corresponds to neither of the first and second windows, then pixel data corresponding to the given pixel is not output to either of the first and second video encoders.

Brief Description of the Drawings

FIG. 1 is a block diagram of a computer system equipped with a stereoscopic display system according to a preferred embodiment of the invention.

FIG. 2 is a block diagram illustrating a preferred internal organization of the frame buffer of FIG. 1.

FIG. 3 a block diagram illustrating the window circuitry of FIG. 1 in more detail.

FIG. 4 is a block diagram illustrating a preferred set of counters located within the stereo output system of FIG. 1.

FIG. 5 is a block diagram illustrating a preferred interface between the stereo output system and the video encoders of FIG. 1.

FIG. 6 is a timing diagram further illustrating the interface shown in FIG. 5.

FIG. 7 is a block diagram illustrating a "vidout" counter located within the stereo output system of FIG. 1.

FIG. 8 is a state diagram illustrating a preferred field pacing control state machine located within the stereo output system of FIG. 1.

FIG. 9 is a state diagram illustrating a preferred field memory write control state machine located within the stereo output system of FIG. 1.

FIG. 10 is a state diagram illustrating preferred video out enable control state machines located within the stereo output system of FIG. 1.

FIG. 11 is a logic diagram illustrating how signals WE_N[0] and WE_N[1] are generated.

Detailed Description of the Preferred Embodiments

FIG. 1 depicts a computer system equipped with a stereoscopic display system according to a preferred embodiment of the invention. Computer system 10 may be any conventional computer system such as a workstation or a personal computer ("PC") having a CPU, memory, disk drive(s) and input devices such as a keyboard and a mouse. Computer system 10 is coupled to optional graphics processing system 12 via bus 14. In most high-end applications, graphics processing system 12 will be present and will include, for example, at least one geometry accelerator and perhaps texture mapping hardware. In such applications, computer system 10 writes polygon data to graphics processing system 12. Graphics processing system 12 rasterizes the polygon data and writes the corresponding pixel data to frame buffer memory 16. In lower-end applications, computer system 10 may write pixel data directly to frame buffer memory 16. In preferred embodiments, frame buffer memory 16 is large enough to accommodate data corresponding to a resolution of either 1280x1024 pixels or 1600x1200 pixels, although more or less memory may be used to accommodate higher or lower resolutions. Digital to analog converter ("DAC") module 18 continually reads the pixel data stored in frame buffer memory 16 and converts the data into corresponding analog signals R, G and B. The RGB signals are used along with horizontal and vertical synchronization signals to drive RGB monitor 20 in a conventional manner. Together, graphics processing system 12, frame buffer memory 16, DAC 18 and RGB monitor 20 comprise a single computer graphics "pipeline."

Computer system 10 is also coupled to window circuitry 22 and stereo output system 24 via bus 14. Pixel data input 25 of stereo output system 24 is coupled via bus 26 to receive pixel data that is exiting frame buffer memory 16 on its way to DAC 18. Pixel data outputs 28, 30 of stereo output system 24 are coupled to pixel data inputs 36, 38 of left and right video encoders 32, 34 as shown. Left and right video encoders 32, 34 have video signal outputs 40, 42. Conventional stereoscopic display device 44 preferably contains two video-format displays 46, 48. Each of the video-format displays 46, 48 is coupled to one of left and right video encoders 40, 42 as shown. Synchronization signals are passed between left and right video encoders 32, 34 as shown at 50. In a preferred embodiment, window circuitry 22, stereo output system 24 and DAC module 18 are formed within a single integrated circuit package 60.

In prior art systems having only one frame buffer, computer system 10 and graphics processing system 12 would fill frame buffer memory 16 once to write one frame of the left image. This left image frame would be displayed, and then computer system 10 and graphics processing system 12 would fill frame buffer memory 16 again to write one frame of the right image. After this right image frame was displayed, the process would be repeated. In an embodiment of the present invention, however, computer system 10 and graphics processing system 12 fill frame buffer memory 16 one time to write one frame of both the left and right images. The left image frame is written into window 52 of frame buffer memory 16, and the right image frame is written into window 54 of frame buffer memory 16. The entire contents of the frame buffer are displayed, and then the process is repeated: On the next fill of frame buffer memory 16, the next frame of the left image is written into window 52, the next frame of the right image is written into window 54, and so on. The result is that both of the windows 52 and 54 are displayed on RGB monitor 20 as shown at 56 and 58. Thus, an observer watching monitor 20 would be able to see both the left and right images simultaneously displayed on the same screen.

As the pixel data is read out of frame buffer memory 16 one line at a time in sequential fashion for display on RGB monitor 20, the primary function of stereo output system 24 is to

pick out which pixel data in each line corresponds to the two windows 52 and 54. Pixel data corresponding to window 52 will be sent to pixel data output 28, while pixel data corresponding to window 54 will be sent to pixel data output 30. In this manner, the contents of window 52 (the left image) will always be displayed on video-format display 48, and the contents of window 54 (the right image) will always be displayed on video-format display 46. Thus, while both the left and right images are displayed simultaneously on RGB monitor 20, the observer wearing stereoscopic display apparatus 44 will see the left image with the left eye and the right image with the right eye. Moreover, the refresh rates for the left and right images of the stereoscopic display are not reduced by 50% as in the prior art systems.

In higher-end embodiments, double buffering may be employed by providing a second frame buffer memory 17. In such an embodiment, windows 52 and 54 would have counterpart windows in frame buffer memory 17. While the left and right image frames contained in windows 52, 54 of frame buffer memory 16 are being displayed, the next left and right image frames may be rendered into the counterpart windows of frame buffer memory 17. When display of the contents of frame buffer memory 16 is complete, the two frame buffer memories may be swapped in a conventional manner so that the contents of frame buffer memory 17 may be displayed while the next left and right image frames may then be rendered into windows 52 and 54, and so on.

Typically for RGB monitors, the contents of frame buffer memory 16 are displayed sequentially one line at a time. For most video-format display devices, however, lines are displayed in interlaced fashion. In other words, all even lines are displayed, and then all odd lines are displayed, and the process is repeated. If video-format displays 46, 48 function in an interlaced mode, a secondary function of stereo output system 24 would be to re-arrange the display lines of pixel data corresponding the left and right images so that they are fed to video encoders 32 and 34 in interlaced fashion (i.e., even lines, then odd lines, etc.) instead of sequentially. A preferred embodiment in which the non-interlaced to interlaced conversion would be required might operate as follows: Given a first complete read by DAC module 18 of all of the scan lines of pixel data in frame buffer memory 16, only even lines would be

captured by stereo output system 24 and the appropriate portions of those lines sent to video encoders 32, 34. Then, on the next complete read by DAC module 18 of all of the scan lines of pixel data in frame buffer memory 16, only odd lines would be captured by stereo output system 24 and the appropriate portions of those lines sent to video encoders 32, 34, and then the process would repeat. In other embodiments, perhaps using a faster dot clock or a different video format, both even and odd lines for each complete scan of frame buffer memory 16 might be captured by stereo output system 24 and output to video encoders 32, 34.

Although scaling and anti-flicker operations are not absolutely necessary for generating left and right images simultaneously from a single frame buffer according to the present invention, in higher-end embodiments a third function of stereo output system 24 would be to perform such scaling and anti-flickering operations on the pixel data to be displayed on stereoscopic display device 44. Preferred methods and apparatus for performing such scaling and anti-flickering operations, as well as for converting non-interlaced pixel data lines into an even/odd interlaced format, are described in detail in copending U.S. patent application serial number 08/626,735, filed April 2, 1996, titled "Video Interface System and Method," which patent application is hereby incorporated by reference in its entirety.

As shown in FIG. 2, frame buffer memory 16 contains n*m pixels, with the (0,0) pixel in the upper leftmost corner and the (n,m) pixel in the lower rightmost corner. The position of window 52 may be designated by the coordinates of its upper leftmost corner pixel, W1(x), W1(y). The position of window 54 may be designated by the coordinates of its upper leftmost corner pixel, W2(x), W2(y). The size of windows 52 and 54 may be designated by the number of pixels in their respective x and y dimensions.

In order for stereo output system 24 to accomplish its primary function of determining which pixel data in the lines read from frame buffer 16 correspond to the left and the right images, three registers are provided in window circuitry 22. As shown in FIG. 3, window circuitry 22 contains a video_out_postion_1 register 62, a video_out_position_2 register 64, and a video_out_size register 66. Software executing in computer system 10 writes data to registers 62, 64 and 66 in order to designate to stereo output system 24 the size and position

of windows 52 and 54. More specifically, the numbers stored in the W1_X_POSITION bit field and the W2_X_POSITION bit field would be the column number for the leftmost pixels in windows 52 and 54, respectively. Similarly, the numbers stored in the W1_Y_POSITION bit field and the W2_Y_POSITION bit field would be the row number for the uppermost pixels in windows 52 and 54, respectively. In a preferred embodiment, both of windows 52 and 54 will be the same size. Thus, only one video_out_size register 66 is needed. The number stored in the X_SIZE bit field would be the number of pixel columns in a window. The number stored in the Y_SIZE bit field would be the number of pixel rows in a window. In other embodiments, more registers may be provided if it becomes desirable to size or proportion windows 52 and 54 differently from each other. It should be noted here that, preferably, the size values placed in these registers will correspond to the size of the windows after any scaling operation is performed. The "e" bits in the two video position registers are enable bits. If zero, stereo output system 24 will not output any pixel data to video encoders 32, 34. If one, stereo output system 24 will be enabled to output pixel data to video encoders 32, 34.

It should be noted that, for a given choice of window size, only one choice will usually be appropriate for the proportions of the window. Those proportions will be dictated by the particular video format, such as NTSC, PAL, SECAM or other, that will be used to drive video-format displays 46 and 48. Moreover, in embodiments utilizing x and y scaling, the appropriate x and y scaling factors to be used by stereo output system 24 in shaping the window images to fit screens 46 and 48 will in turn be dictated by the size and proportions chosen for the windows. Thus, for example, if a 2:1 scale factor were used for x and y and the NTSC format were desired at the output of video encoders 32 and 34, the actual size of windows 52 and 54 in frame buffer memory 16 might be 1280x960 pixels, while the x and y size values stored in video_out_size register 66 might be X_SIZE=640 and Y_SIZE=480 pixels.

FIG. 4 shows a preferred set of counters used to track the clocking of pixel data as it exits frame buffer memory 16 and is clocked into DAC module 18. For window 52, x offset counter 400, y offset counter 402, x size counter 404 and y size counter 406 are provided. For

window 54, x offset counter 408, y offset counter 410, x size counter 412 and y size counter 414 are provided. In operation, x offset counter 400 is loaded with an offset value corresponding to W1_X_POSITION; y offset counter 402 is loaded with an offset value corresponding to W1_Y_POSITION; x size counter 404 is loaded with a value corresponding to X_SIZE; and y size counter 406 is loaded with a value corresponding to Y_SIZE. Similarly, x offset counter 408 is loaded with an offset value corresponding to W2_X_POSITION; y offset counter 410 is loaded with an offset value corresponding to W2_Y_POSITION; x size counter 412 is loaded with a value corresponding to X_SIZE; and y size counter 414 is loaded with a value corresponding to Y_SIZE.

The x offset counters are reset on a line-by-line basis, and are decremented once for each pixel clocked out of frame buffer memory 16. The y offset counters are reset on a frame-by-frame basis, and are decremented once for each line of pixels clocked out of frame buffer memory 16. Each of the counters is equipped with a comparison function, as indicated by the blocks containing the subtraction and equality symbols, and outputs a "done" signal when the respective counts have been reached. (In the drawing, the outputs are shown as "early done" signals for timing considerations, and are generated one clock before the respective counts have been reached.) As will be described in more detail below, the done signals are used to determine whether pixel data present on bus 26 corresponds to left image window 52, right image window 54 or neither.

FIG. 5 is a block diagram illustrating a preferred interface between stereo output system 24 and video encoders 32, 34. As shown, each of video encoders 32, 34 contains a field memory bank 0 and a field memory bank 1. The two field memory banks in each video encoder comprise a FIFO for pixel data entering the encoder. For interlaced display formats, even scan lines are written into the FIFOs first by stereo output system 24, followed by odd scan lines (as described above). Twenty-four bit data bus VO_DATA[23:0] is coupled to both video encoders 32, 34. Likewise, write clocks WCK[0] and WCK[1] as well as reset line WRST_N are coupled to both encoders. WCK[0] is the write clock for field memory banks 0, and WCK[1] is the write clock for field memory banks 1. WRST_N resets the write pointer in all

field memories to the first position of the FIFO. WE_N[0] is coupled only to video encoder 34, while WE_N[1] is coupled only to video encoder 32. These are the enable signals that control writes of pixel data over bus VO_DATA into the field memories corresponding to video encoders 34 and 32, respectively. The VO_FIELD signal indicates which field is currently being displayed by video encoders 34, 32, and is used by stereo output system 24 to control the timing of pixel data writes to video encoders 34, 32 (so as not to write odd lines while odd lines are being displayed and not to write even lines while even lines are being displayed).

For the sake of the discussion that follows, the rate at which pixels are read out of frame buffer memory 16 by DAC module 18 will be referred to as the "dot clock" rate. In a an embodiment, the rate at which pixels are sent by stereo output system 24 to video encoders 32, 34 (writing to one video encoder at a time) is one-fourth of the dot clock rate. Even pixels from each line are written to field memory banks 0, while odd pixels are written to field memory banks 1. This is accomplished by making WCK[0] and WCK[1] 180 degrees out of phase with one another (as shown in FIG. 6), and by setting their frequencies to be one-eighth of the dot clock frequency. In this manner, the field memory writes are able to keep pace with the dot clock. For interlaced video formats, the rate disparity between the dot clock and the pixel output rate of stereo output system 24 is made less problematic (by a factor of 2) by capturing only even or odd lines for each scan of frame buffer memory 16 (as described above). In addition, because windows 52 and 54 are typically not as long in the x direction as is the frame buffer, additional "catch up" time is available for stereo output system 24 because the latter system must only output pixels corresponding to the windows. Finally, a buffer may be used in stereo output system 24 to resolve any remaining rate disparity problems. In one embodiment, a FIFO buffer was provided that was capable of buffering up to 1024 pixels (each having 24 bits of color information associated with it).

Beyond the above-described details of the interface with stereo output system 24, each of video encoders 32, 34 may be implemented in a conventional manner and may be based, for example, on an off-the-shelf component such as an SAA7199B digital video encoder chip

manufactured and sold by Philips Semiconductors, Inc. As indicated in FIG. 1 at bus 50, the SAA7199B chips may be caused to operate in synch with one another using slave mode and routing synch outputs from one video encoder to the synch inputs of the other video encoder. This is preferable in that it causes the refresh cycle in both video-format displays 46 and 48 to be synchronized with one another.

The state machines and control blocks within stereo output system 24 will now be described in detail with reference to FIGs 7-11.

Video Out Control Block

The video out control block comprises two state machines: a field pacing control state machine and a field memory write control state machine. The video out control block also makes use of a video out counter as shown in FIG. 7.

Field Pacing Control State Machine. The function of the field pacing control state machine is to control the output of even or odd lines of pixel data to video encoders 32, 34. When the encoders are displaying the odd field, the state machine signals that even lines can be written to the field memories, and vice versa. This pacing of data writes is done to minimize "tearing" artifacts in the displayed images. Any suitable logic mechanism can be used to implement the field pacing control state machine. In a preferred embodiment, the field pacing control state machine is implemented according to the state diagram shown in FIG. 8. Note: state transitions for the state machine shown in FIG. 8 occur at the beginning of each frame of data read from frame buffer memory 16 (i.e., at posedge of vblank_n). State transitions for all other state machines occur in synchronization with the dot clock. Signal descriptions for the signals shown in the state diagram are shown in Table A below:

Signal Name	Description
vop_e	The enable bit of the video_out_position_1 register.
vop_e1	The enable bit of the video_out_position_2 register.
CurrentField	This is the VO_FIELD signal, latched in synchronization with the dot clock.
voc_update	When set, field pacing is disabled. (This is used for test purposes only.)
OutputField	Indicates whether even or odd lines of the frame currently being read from frame buffer memory 16 will be used to send pixel data to video encoders 32, 34.
OutputEnable	Indicates that pixel data in the frame currently being read from frame buffer memory 16 will be used to send pixel data to video encoders 32, 34.
st_wrst_n	Logically identical to WRST_N.
posedge(st_vblank_n)	Positive edge of active low signal that indicates vertical blank.
posedge(st_hblank_n)	Positive edge of active low signal that indicates horizontal blank.

Table A

Field Memory Write Control State Machine. The function of the field memory write control state machine is to generate the write clocks WCK[1:0] and to generate the global write enable signal WriteEnable_n, which is gated with the WindowState signals from the video out enable control block to generate WE_N[0] and WE_N[1]. Any suitable logic mechanism can be used to implement the field memory write control state machine. In a preferred embodiment, the field memory write control state machine is implemented according to the state diagram shown in FIG. 9. Signal descriptions for the signals shown in the state diagram are shown in Table B below:

Signal Name	Description
pre_st_wck[1:0]	Logically identical to WCK[1:0]
WriteEnable_n	A low true signal indicating that data should be written to one of the video encoders.
RequestVideoPixel	Causes vidout counter to be decremented and a pixel to be read out of the line buffer.
sampled_WriteEnable_n	A one-clock state delayed version of WriteEnable_n.
crc_enable	In an embodiment using conventional cyclic redundancy check techniques for test purposes, this signal, when asserted, causes the CRC to accumulate.
EarlyEarlyVidOutDone	Asserted when vidout counter value is equal to 2.
VideoOutReady	Indicates data in the line buffer is ready to be output. Causes the vidout counter to be loaded with X_SIZE (also denoted vos_x).
VidOutDone	Indicates that the last pixel in a line corresponding to window 52 or 54 has been clocked out to a video encoder.
EnableCRC	In an embodiment using conventional cyclic redundancy check techniques for test purposes, this signal indicates to the field memory write control state machine that CRC computations are enabled and that it may enable writes to a CRC accumulation register.

Table B

Video Out Enable Control Block

The function of the video out enable control block is to generate the write enable signals shown in FIG. 5. Any suitable logic mechanism can be used to implement the video out enable control block. In a preferred embodiment, it comprises the two state machines shown in FIG. 10 and the logic in FIG. 11. Signal descriptions for the signals shown in the state diagram are shown in Table C below:

Signal Name	Description
YsizeDone	1 indicates the value in Y size counter is 0.
YSizeDone1	1 indicates the value in Y size counter 1 is 0.
YFilterIdle	1 indicates the y filter is idle.
VidOutDone	1 indicates the value in vidout counter is 0.
st_vblank_n	Active low signal indicating vertical blank.
YoffsetDone	1 indicates the value in Y offset counter is 0.
YOffsetDone1	1 indicates the value in Y offset counter 1 is 0.
vop_e	(See above.)
vop_e1	(See above.)
WindowState0	Combined with global WriteEnable_n to produce WE_N[0].
WindowState1	Combined with global WriteEnable_n to produce WE_N[1].

Table C

Window Position Control Block

The window position control block generates the counter signals shown in FIG. 4, and also controls the loading and decrementing of the counters. Any suitable logic mechanism may be used to implement the window position control block. In a preferred embodiment, it was implemented according the following VERILOG code:

```
//=============================================================================
// Counter Done Signals
//=============================================================================
reg                     XOffsetDone;
reg                     YOffsetDone;
reg                     XSizeDone;
reg                     YSizeDone;
reg                     XOffsetDone1;
reg                     YOffsetDone1;
reg                     XSizeDone1;
reg                     YSizeDone1;
reg                     VidOutDone;
reg                     EarlyVidOutDone;
reg                     EarlyEarlyVidOutDone;
reg                     VidOutTileDone;

always @(negedge dot_clk_n) begin
    //=================================================
    // Window 0 Counters
    //=================================================
    if (dec_XOffsetCtr)
        XOffsetDone <= earlyXOffsetDone;
    else if (set_XOffsetCtr)
        XOffsetDone <= vop_x_zero;

    if (dec_YOffsetCtr)
        YOffsetDone  <= earlyYOffsetDone;
    else if (set_YOffsetCtr)
        YOffsetDone <= vop_y_zero;

    if (dec_XSizeCtr)
        XSizeDone <= earlyXSizeDone;
```

```
    else if (set_XSizeCtr)
        XSizeDone <= vos_x_zero;

    if (dec_YSizeCtr)
        YSizeDone <= earlyYSizeDone;
    else if (set_YSizeCtr)
        YSizeDone <= vos_y_zero;

    //================================================
    // Window 1 Counters
    //================================================
    if (dec_XOffsetCtr1)
        XOffsetDone1 <= earlyXOffsetDone1;
    else if (set_XOffsetCtr1)
        XOffsetDone1 <= vop1_x_zero;

    if (dec_YOffsetCtr1)
        YOffsetDone1 <= earlyYOffsetDone1;
    else if (set_YOffsetCtr1)
        YOffsetDone1 <= vop1_y_zero;

    if (dec_XSizeCtr1)
        XSizeDone1 <= earlyXSizeDone1;
    else if (set_XSizeCtr1)
        XSizeDone1 <= vos_x_zero;

    if (dec_YSizeCtr1)
        YSizeDone1 <= earlyYSizeDone1;
    else if (set_YSizeCtr1)
        YSizeDone1 <= vos_y_zero;

    //================================================
    // Video Out Counter
    //================================================
    if (st_reset_n == 0) begin
        EarlyEarlyVidOutDone <= 0;
        EarlyVidOutDone <= 0;
        VidOutDone <= 0;
    end
    else if (dec_VidOutCtr) begin
        EarlyEarlyVidOutDone <= EarlyEarlyEarlyVidOutDone;
        EarlyVidOutDone <= EarlyEarlyVidOutDone;
        VidOutDone <= EarlyVidOutDone;
    end
    else if (set_VidOutCtr)
        VidOutDone <= vos_x_zero;

    //================================================
    // Video Out Tile Done
    //================================================
    VidOutTileDone <= VidQutDone
                    | EarlyVidOutDone
                    | EarlyEarlyVidOutDone
                    | EarlyEarlyEarlyVidOutDone;
end

//================================================================================
// Priority encode set_XOffsetCtr & dec_XOffsetCtr
//================================================================================
reg                 set_XOffsetCtr;
reg                 dec_XOffsetCtr;

always @(st_reset_n or pos_hsync_n or st_hblank_n or XOffsetDone or
         YOffsetDone or YSizeDone) begin
```

```
    if (st_reset_n == 0) begin
        set_XOffsetCtr = 1;
        dec_XOffsetCtr = 0;
    end
    //=======================================
    // On pos edge of hsync_n preload XOffset
    //=======================================
    else if (pos_hsync_n) begin
        set_XOffsetCtr = 1;
        dec_XOffsetCtr = 0;
    end
    //=======================================
    // When active video and YOffset done and
    // XOffset not done decrement XOffset
    //=======================================
    else if (st_hblank_n && !XOffsetDone && YOffsetDone && !YSizeDone) begin
        set_XOffsetCtr = 0;
        dec_XOffsetCtr = 1;
    end
    else begin
        set_XOffsetCtr = 0;
        dec_XOffsetCtr = 0;
    end
end

//=============================================================================
// Priority encode set_YOffsetCtr & dec_YOffsetCtr
//=============================================================================
reg                 set_YOffsetCtr;
reg                 dec_YOffsetCtr;

always @(st_reset_n or pos_vblank_n or st_vblank_n or neg_hblank_n or
         YOffsetDone) begin
    if (st_reset_n == 0) begin
        set_YOffsetCtr = 1;
        dec_YOffsetCtr = 0;
    end
    //=======================================
    // On pos edge of vblank_n preload YOffset
    //=======================================
    else if (pos_vblank_n) begin
        set_YOffsetCtr = 1;
        dec_YOffsetCtr = 0;
    end
    //=======================================
    // When active video and YOffset not done
    // and negedge hblank_n decrement YOffset
    //=======================================
    else if (st_vblank_n && neg_hblank_n && !YOffsetDone) begin
        set_YOffsetCtr = 0;
        dec_YOffsetCtr = 1;
    end
    else begin
        set_YOffsetCtr = 0;
        dec_YOffsetCtr = 0;
    end
end

//=============================================================================
// Priority encode set_XSizeCtr & dec_XSizeCtr
//=============================================================================
```

```
reg                 set_XSizeCtr;
reg                 dec_XSizeCtr;

always @(st_reset_n or pos_hsync_n or XFPixelValid or XSizeDone or vop_e) begin
    if (st_reset_n == 0) begin
        set_XSizeCtr = 1;
        dec_XSizeCtr = 0;
    end
    //=======================================
    // On pos edge of hsync_n preload XSize
    //=======================================
    else if (pos_hsync_n) begin
        set_XSizeCtr = 1;
        dec_XSizeCtr = 0;
    end
    //=======================================
    // When XFilter outputs a pixel dec XSize
    //=======================================
    else if (XFPixelValid && !XSizeDone && vop_e) begin
        set_XSizeCtr = 0;
        dec_XSizeCtr = 1;
    end
    else begin
        set_XSizeCtr = 0;
        dec_XSizeCtr = 0;
    end
end

//=============================================================================
// Priority encode set_YSizeCtr & dec_YSizeCtr
//=============================================================================
reg                 set_YSizeCtr;
reg                 dec_YSizeCtr;

always @(st_reset_n or pos_vblank_n or YFLineDone or YOffsetDone or YSizeDone or vop_e) begin
    if (st_reset_n == 0) begin
        set_YSizeCtr = 1;
        dec_YSizeCtr = 0;
    end
    //=======================================
    // On pos edge of vblank_n preload YSize
    //=======================================
    else if (pos_vblank_n) begin
        set_YSizeCtr = 1;
        dec_YSizeCtr = 0;
    end
    //=======================================
    // When YFilter outputs a line dec YSize
    //=======================================
    else if (YFLineDone && YOffsetDone && !YSizeDone && vop_e) begin
        set_YSizeCtr = 0;
        dec_YSizeCtr = 1;
    end
    else begin
        set_YSizeCtr = 0;
        dec_YSizeCtr = 0;
    end
end

//=============================================================================
// Priority encode set_XOffsetCtr1 & dec_XOffsetCtr1
//=============================================================================
```

```
reg                     set_XOffsetCtrl;
reg                     dec_XOffsetCtrl;

always @(st_reset_n or pos_hsync_n or st_hblank_n or XOffsetDone1 or
         YOffsetDone1 or YSizeDone1) begin
    if (st_reset_n == 0) begin
        set_XOffsetCtrl = 1;
        dec_XOffsetCtrl = 0;
    end
    //=======================================
    // On pos edge of hsync_n preload XOffset
    //=======================================
    else if (pos_hsync_n) begin
        set_XOffsetCtrl = 1;
        dec_XOffsetCtrl = 0;
    end
    //=======================================
    // When active video and YOffset done and
    // XOffset not done decrement XOffset
    //=======================================
    else if (st_hblank_n && !XOffsetDone1 && YOffsetDone1 && !YSizeDone1) begin
        set_XOffsetCtrl = 0;
        dec_XOffsetCtrl = 1;
    end
    else begin
        set_XOffsetCtrl = 0;
        dec_XOffsetCtrl = 0;
    end
end

//=============================================================================
// Priority encode set_YOffsetCtrl & dec_YOffsetCtrl
//=============================================================================
reg                     set_YOffsetCtrl;
reg                     dec_YOffsetCtrl;

always @(st_reset_n or pos_vblank_n or st_vblank_n or neg_hblank_n or
         YOffsetDone1) begin
    if (st_reset_n == 0) begin
        set_YOffsetCtrl = 1;
        dec_YOffsetCtrl = 0;
    end
    //=======================================
    // On pos edge of vblank_n preload YOffset
    //=======================================
    else if (pos_vblank_n) begin
        set_YOffsetCtrl = 1;
        dec_YOffsetCtrl = 0;
    end
    //=======================================
    // When active video and YOffset not done
    // and negedge hblank_n decrement YOffset
    //=======================================
    else if (st_vblank_n && neg_hblank_n && !YOffsetDone1) begin
        set_YOffsetCtrl = 0;
        dec_YOffsetCtrl = 1;
    end
    else begin
        set_YOffsetCtrl = 0;
        dec_YOffsetCtrl = 0;
    end
end
```

```
//=============================================================================
// Priority encode set_XSizeCtrl & dec_XSizeCtrl
//=============================================================================
reg                     set_XSizeCtrl;
reg                     dec_XSizeCtrl;

always @(st_reset_n or pos_hsync_n or XFPixelValid or XSizeDone1 or
        vop1_e) begin
    if (st_reset_n == 0) begin
        set_XSizeCtrl = 1;
        dec_XSizeCtrl = 0;
    end
    //=======================================
    // On neg edge of hblank_n preload XSize
    //=======================================
    else if (pos_hsync_n) begin
        set_XSizeCtrl = 1;
        dec_XSizeCtrl = 0;
    end
    //=======================================
    // When XFilter outputs a pixel dec XSize
    //=======================================
    else if (XFPixelValid && !XSizeDone1 && vop1_e) begin
        set_XSizeCtrl = 0;
        dec_XSizeCtrl = 1;
    end
    else begin
        set_XSizeCtrl = 0;
        dec_XSizeCtrl = 0;
    end
end

//=============================================================================
// Priority encode set_YSizeCtrl & dec_YSizeCtrl
//=============================================================================
reg                     set_YSizeCtrl;
reg                     dec_YSizeCtrl;

always @(st_reset_n or pos_vblank_n or YFLineDone or YOffsetDone1 or YSizeDone1 or vop1_e) begin
    if (st_reset_n == 0) begin
        set_YSizeCtrl = 1;
        dec_YSizeCtrl = 0;
    end
    //=======================================
    // On pos edge of vblank_n preload YSize
    //=======================================
    else if (pos_vblank_n) begin
        set_YSizeCtrl = 1;
        dec_YSizeCtrl = 0;
    end
    //=======================================
    // When YFilter outputs a line dec YSize
    //=======================================
    else if (YFLineDone && YOffsetDone1 && !YSizeDone1 && vop1_e) begin
        set_YSizeCtrl = 0;
        dec_YSizeCtrl = 1;
    end
    else begin
        set_YSizeCtrl = 0;
        dec_YSizeCtrl = 0;
    end
end
```

```
//=============================================================================
// Priority encode set_VidOutCtr & dec_VidOutCtr
//=============================================================================
reg                     set_VidOutCtr;
reg                     dec_VidOutCtr;
reg                     delayed_VideoOutReady;

always @(negedge dot_clk_n) begin
    delayed_VideoOutReady <= VideoOutReady;
end

always @(st_reset_n or VideoOutReady or delayed_VideoOutReady or
         RequestVideoPixel or VidOutDone) begin
    if (st_reset_n == 0) begin
        set_VidOutCtr = 1;
        dec_VidOutCtr = 0;
    end
    //=======================================
    // On rising edge of VideoOutReady
    //=======================================
    else if (VideoOutReady && !delayed_VideoOutReady) begin
        set_VidOutCtr = 1;
        dec_VidOutCtr = 0;
    end
    //=======================================
    // When pixel has been shipped out dec
    // count
    //=======================================
    else if (RequestVideoPixel && !VidOutDone) begin
        set_VidOutCtr = 0;
        dec_VidOutCtr = 1;
    end
    else begin
        set_VidOutCtr = 0;
        dec_VidOutCtr = 0;
    end
end
```

While the invention has been described in detail in relation to particular embodiments thereof, this description is intended to be illustrative only. It will be obvious to those skilled in the art that many modifications can be made to the described embodiments without departing from the spirit and scope of the invention, and that such modifications will remain within the scope of the following claims.

What is claimed is:

1. Apparatus for generating left and right video channels to drive a stereoscopic display device 44 using only one frame buffer memory 16 of a computer graphics pipeline, said apparatus comprising:

window circuitry 22 for defining first and second windows 52, 54 within said frame buffer memory 16, said first and second windows 52, 54 for storing pixel data corresponding to left and right views, respectively, of a stereoscopic image;

first and second video encoder systems 32, 34, each having a pixel data input 36, 38 and a video signal output 40, 42, and each operable to present a video format signal at its video signal output responsive to pixel data presented to its pixel data input; and

a stereo output system 24 coupled to said frame buffer memory 16, said window circuitry 22 and said first and second video encoder systems 32, 34, said stereo output system 24 operable to present pixel data read from said first window 52 within said frame buffer memory 16 to the pixel data input 36 of said first video encoder system 32, and to present pixel data read from said second window 54 within said frame buffer memory 16 to the pixel data input 38 of said second video encoder system 34;

the video format signals presented at the video signal outputs 40, 42 of said first and second video encoder systems 32, 34 comprising said left and right video channels.

2. The apparatus of claim 1, wherein said window circuitry 22 comprises:

first and second video position registers 62, 64 for storing data that indicates the position of said first and second windows 52, 54, respectively, within said frame buffer memory 16; and

a video size register 66 for storing data that indicates the size of said first and second windows 52, 54.

3. The apparatus of claim 1, wherein said stereo output system comprises at least one counter 400, 402, 404, 406, 408, 410, 412, 414 operable to track the X and Y coordinates of pixels as they are read out of said frame buffer memory 16.

4. The apparatus of claim 2, wherein:

said first and second video position registers 62, 64 are each operable to store an X offset value and a Y offset value, set X and Y offset values corresponding to the coordinates of the upper leftmost corner of said first and second windows 52, 54, respectively; and

wherein said video size register 66 is operable to store an X size value and a Y size value, said X and Y size values corresponding to the number of columns and rows in said first and second windows 52, 54.

5. The apparatus of claim 4, wherein:

said stereo output system comprises at least one counter 400, 402, 404, 406, 408, 410, 412, 414 operable to track the X and Y coordinates of pixels as they are read out of said frame buffer memory 16; and

wherein said stereo output system is also operable to compare the state of said at least one counter 400, 402, 404, 406, 408, 410, 412, 414 with the values stored in said first and second video position registers 62, 64 and said video size register 66, and to select pixel data for output to said first and second video encoder systems 32, 34 responsive to the comparison.

6. A method of generating left and right video channels to drive a stereoscopic display device 44 using only one frame buffer memory 16 of a computer graphics pipeline, wherein each memory location for storing pixel data within said frame buffer memory 16 has an X and a Y position coordinate, said method comprising the steps of:

allocating first and second windows 52, 54 within said frame buffer memory 16;

storing position and size information 62, 64, 66, said size information 66 corresponding to the size of said first and second allocated windows and said position information 62, 64 corresponding to the positions of said first and second allocated windows 52, 54 within said frame buffer memory 16;

writing pixel data corresponding to a left image into said first window 52;

writing pixel data corresponding to a right image into said second window 54;

reading pixel data out of said frame buffer memory 16 serially;

keeping an X count and a Y count corresponding to the coordinates of the pixels as they read out of said frame buffer memory 16;

comparing said X count and said Y count for a given pixel with said position and size information 62, 64, 66; and

based on the outcome of said comparing step, determining whether said given pixel corresponds to said first window 52, said second window 54, or neither of said first or second windows 52, 54.

7. The method of claim 6, further comprising the steps of:

if it is determined in said determining step that said given pixel corresponds to said first window 52, outputting pixel data corresponding to said given pixel to a first video encoder 32; else

if it is determined in said determining step that said given pixel corresponds to said second window 54, outputting pixel data corresponding to said given pixel to a second video encoder 34; else

if it is determined in said determining step that said given pixel corresponds to neither of said first and second windows 52, 54, not outputting pixel data corresponding to said given pixel to either of said first and second video encoders 32, 34.

The Patent Office

Application No:	GB 9806180.7	**Examiner:**	Sue Willcox
Claims searched:	1, 6	**Date of search:**	23 July 1998

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.P): H4F FDD; H4T TBAS

Int Cl (Ed.6): H04N 13/00; G06T (15/00, 17/00)

Other: Online database: WPI

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
A	GB 2312122 A (IBM)	
A	GB 2308284 A (Mitsubishi Denki)	

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.